UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/616,525	07/10/2003	Mark Robert Funk	ROC920020205US1	1216

7590 05/28/2008

Grant A. Johnson
IBM Corporation - Dept. 917
3605 Highway 52 North
Rochester, MN 55901

EXAMINER	
RUTTEN, JAMES D	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
05/28/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 10/616,525
Filing Date: July 10, 2003
Appellant(s): FUNK ET AL.

Joan Pennington, Reg. No. 30,885
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed 4/24/08 appealing from the Office action mailed 12/28/07.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

Bickle et al, "Differential Effective Lapse Time Accumulator (Delta)" IBM Technical Disclosure Bulletin, vol. 27, Issue 2 (July 1, 1984), pp. 1258-1259

Rosenberg, "How Debuggers Work" John Wiley & Sons, (1996) Chapters 1-3

5,657,253 DREYER et al. 08-1997

6,249,907	CARTER et al.	06-2001
5,533,192	HAWLEY et al.	07-1996

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

- Claims 1, 2, 4-6, 8-11, 13, and 14 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record "Differential Effective Lapse Time Accumulator (Delta)" by Bickle et al. (hereinafter "Bickle") in view of prior art of record "How Debuggers Work" by Rosenberg (hereinafter "Rosenberg") in view of prior art of record U.S. Patent 5,657,253 to Dreyer et al. (hereinafter "Dreyer") in view of U.S. Patent 6,249,907 to Carter et al. (hereinafter "Carter").

In regard to claim 1, Bickle discloses:

A method for implementing breakpoint based performance measurement using a set of hardware counters for counting hardware events See page 1, e.g. "time/counter cards 24"*; said hardware counters being programmable for counting predefined ... processor events* See page 1 e.g. "measurement of system performance"*; said predefined ... processor events including processor cycles ...* See page 1, e.g. "instruction cycle time measurement"; Bickle does not expressly disclose: *programmable processor events including ... cache misses.* However, Dreyer teaches that events are programmable, and

include cache misses. See column 3 lines 52-65, e.g. "programmable event counters" also "cache miss rates."

Bickle discloses *a start breakpoint instruction and a stop breakpoint instruction;* See Bickle middle of page 1, e.g. "start breakpoint A and stop breakpoint B"; Bickle does not expressly disclose: *providing compiler-generated hardware instructions defining breakpoint instructions within an instruction stream; said compiler-generated hardware instructions including [a start breakpoint instruction and a stop breakpoint instruction;]* However, Carter teaches that compilers generate hardware instructions defining breakpoints within an instruction stream. See column 6 lines 44-48, e.g. "cause the compiler to generate debugger hook functions calls…" Note that compilers function to translate high level program code into low level machine, or "object code." See column 5 lines 31-32. As such, Carter's "hook functions" are implemented as hardware instructions.

inserting said start breakpoint instruction and said stop breakpoint instruction…; See Bickle middle of page 1, e.g. "start breakpoint A and stop breakpoint B"; Bickle does not expressly disclose: inserting breakpoint *instructions…in* [compiler generated] *hardware instructions for a user source code*. However, Rosenberg teaches that breakpoints can be implemented as hardware instructions for a user source code. See bottom of page 40, e.g. "special instruction". Also see page 24, e.g. "Source View." Bickle does not expressly disclose *executing said* [compiler-generated] *hardware instructions and suspending processing of said hardware instructions responsive to executing said start breakpoint instruction;* However, Rosenberg teaches that upon

encountering a "special instruction," execution is suspended while an operating system notifies a debugger. See bottom of page 40.

responsive to executing said start breakpoint instruction generating a processor interrupt...; See page 1 line 21, e.g. "The A comparator 18 is used as a start timing breakpoint..." Comparators are used to provide a signal (i.e. interrupt) to the accumulator. Bickle does not expressly disclose *for entering interrupt handler instructions and calling breakpoint instructions;* However, Rosenberg teaches that upon encountering a "special instruction," a trap to the operating system is called which notifies a debugger, i.e. "breakpoint instructions". See bottom of page 40. Note that the limitation "calling breakpoint instructions" is broadly interpreted according to the description providing enablement on page 4 lines 11-21 which describes a "debugger."

...starting said defined set of hardware counters; See page 1 lines 26-27, e.g. "accumulate elapsed time"; Bickle does not expressly disclose *said breakpoint instructions generating a start processing instruction to return processing from said interrupt handler instructions to* [said compiler-generated] *hardware instructions..., responsive to said generated start processing instruction;* However, Rosenberg teaches that a debugger handles a breakpoint before returning execution. See page 41 lines 16-17, e.g. "proceed past this breakpoint." Further, see "Algorithm 3.1 appearing on page 42, e.g. "run debuggee full speed."

executing the [compiler-generated] *hardware instructions and ...and stopping said defined set of hardware counters, responsive to executing said stop breakpoint instruction.* See page 1 lines 22-23, e.g. "B comparator 18 is used as a stop timing

breakpoint." Bickle does not expressly disclose *suspending processing of the* [compiler generated] *hardware instructions.* As pointed out above, Rosenberg teaches breakpoint handling by suspending processing. See bottom of page 4, e.g. "trap to the operating system."

providing a debugger breakpoint manager including a performance measurement program and a user interface, and enabling a user to specify a start bound and an end bound of a performance collection region of said user source code and said set of hardware counters. See page 1 lines 33-35. Here, Bickle's "operator interface" coordinates with the test tool to display performance results, and at least provides a breakpoint manager where users can enter breakpoint parameters (including start and end bounds provided by the A and B comparators– see lines 21-23). These breakpoint parameters control the region over which the hardware counters (i.e. input timer/counter cards) operate.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Dreyer's programmable counter with Bickle's event counting in order to monitor particular aspects of processor performance as suggested by Dreyer (see column 3 lines 60-65). Further, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use Rosenberg's method of breakpoint handling with Bickle's breakpoints in order to provide control over the execution of a debuggee (see Rosenberg page 39 paragraph 1). Finally, it would have been obvious to one of ordinary skill in the art at the time the invention was made to use Carter's hook

functions with Bickle's breakpoints in order to enable a debugger to stop execution as suggested by Carter (see column 6 lines 42-43).

In regard to claim 2, the above rejection of claim 1 is incorporated. Bickle further discloses: *wherein said predefined processor events further include at least one of processor instructions executed, a defined type of processor instruction executed, and translation lookaside buffer misses.* See page 1 lines 28-29, e.g. "number of times breakpoint A occurs." Note that the phrase "at least one of..." permits the disclosure of one item to meet the language of the claim.

In regard to claim 4, the above rejection of claim 1 is incorporated. Bickle further discloses: *wherein the inserting step includes inserting said start breakpoint instruction and said stop breakpoint instruction at arbitrary user defined locations ....* See page 1 line 34, e.g. "breakpoint ... parameters." Bickle does not expressly disclose *in said hardware instructions.* However, Rosenberg teaches that breakpoints are inserted in hardware instructions. See page 41 lines 5-6. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Rosenberg's method of breakpoint handling with Bickle's breakpoints in order to provide control over the execution of a debuggee (see Rosenberg page 39 paragraph 1).

In regard to claim 5, the above rejection of claim 1 is incorporated. Bickle further discloses: *a user.* See page 1 line 1, e.g. "user." Bickle does not expressly disclose:

enabling ... to interrogate a program state and to request said start processing instruction. However, Rosenberg teaches that a debugger interrogates program state and enables a return to program processing. See page 41 lines 13-20. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Rosenberg's method of breakpoint handling with Bickle's breakpoints in order to provide control over the execution of a debuggee (see Rosenberg page 39 paragraph 1).

In regard to claim 6, Bickle discloses:

Apparatus for implementing breakpoint based performance measurement See page 1 lines 13-18, e.g. "DELTA system."

...a breakpoint manager; See page 1 lines 31-35. Here, the "operator interface" shows the existence of a breakpoint manager. That is, the breakpoint manager operates to manage breakpoints using input provided by the operator interface.

said breakpoint manager utilizing said performance measurement program and said user interface for defining a set of said hardware counters for counting user specified hardware events *See page 1, lines 1-3, e.g. "allows the user to take accurate time measurements" and "count the number of times." Also lines 33-35, e.g. "operator interface."*

user program means See page 1 line 1, e.g. "test tool."

Bickle does not expressly disclose: *a source level debugger including a breakpoint manager;* However, Rosenberg teaches that a source level debugger (see page 4 line 2, e.g. "source-level debugger") is used to manage breakpoints (see page 5, 3rd paragraph, e.g. "Current debuggers can control all execution ... by using breakpoints"). It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Rosenberg's debugger with Bickle's breakpoints in order to provide control over the execution of a debuggee (see Rosenberg page 5 paragraph 3).

All further limitations have been addressed in the above rejection of claim 1.

In regard to claim 8, the above rejection of claim 6 is incorporated. Bickle further discloses: *wherein said breakpoint manager ... records user information specifying said defined set of hardware counters*. See page 1 lines 33-35. Bickle does not expressly disclose *responsive to said start breakpoint instruction*. However, Rosenberg teaches that information can be recorded responsive to a breakpoint. See page 41 lines 13-16. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Rosenberg's information recording with Bickle's user information in order to give a programmer fine control over a program (see Rosenberg, top of page 3).

In regard to claims 9 and 10, the above rejection of claim 6 is incorporated. All further limitations have been addressed in the above rejection of claims 2 and 4, respectively.

In regard to claim 11, Bickle discloses a computer program product. See page 1 line 1, e.g. "DELTA is a test tool." All further limitations have been addressed in the above rejection of claim 1.

In regard to claims 13-14, the above rejection of claim 11 is incorporated. All further limitations have been addressed in the above rejection of claims 4 and 2, respectively.

- Claim 7 is rejected under 35 U.S.C. 103(a) as being unpatentable over Bickle, Rosenberg, Dreyer, and Carter as applied to claim 6 above, and further in view of U.S. Patent No. 5,533,192 to Hawley et al. (hereinafter "Hawley").

In regard to claim 7, the above rejection of claim 6 is incorporated. Bickle further discloses: *specifying said defined set of hardware counters*. See page 1 lines 25-29. Bickle, Rosenberg, and Dreyer do not expressly disclose *wherein start breakpoint instruction includes encoded information*. However, Hawley teaches that breakpoints are encoded to indicate the type of breakpoint as well as the identity of the desired debugger. See column 9 lines 24-28. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Hawley's teaching of breakpoint encoding with Bickle's hardware counters in order to provide more than one debugger operative at a time (see Hawley column 5 lines 40-42).

(10) Response to Argument

Appellants' arguments filed 4/24/08 have been fully considered but they are not persuasive.

Appellants' arguments generally fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without *specifically* pointing out how the language of the claims patentably distinguishes them from the references. For example, at the bottom of page 27 of the brief, continuing to page 28, Appellants state:

> Bickle, Rosenberg, Dreyer et al., Carter et al., and Hawley references fail to suggest inserting a start breakpoint instruction and a stop breakpoint instruction in hardware instructions; executing said hardware instructions and suspending processing of said hardware instructions responsive to

> executing said start breakpoint instruction; and responsive to executing said start breakpoint instruction generating a processor interrupt for entering interrupt handler instructions and calling breakpoint instructions, and that said breakpoint instructions generating a start processing instruction to return processing from said interrupt handler instructions to the hardware instructions and starting said defined set of hardware counters, responsive to said generated start processing instruction, as taught and claimed by applicants. [emphasis in original]

In this particular passage, no specific prior art reference is addressed and no explanation was found which specifically points out how the language of the claims patentably distinguishes them from the references. Such allegations are not persuasive enough to overcome the prima facie rejections of the claims. Indeed, such arguments do not address the technical analysis of the prima facie rejection in a manner that specifically distinguishes the claims from the references. While certain portions of the passage were underlined, no further explanation was provided in support of the claim. Similar allegations of patentability are found throughout the remainder of Appellants' arguments on pages 29-44 of the brief. For example, on page 29 of the brief, Appellants argue:

> Applicants respectfully submit that the combined teachings of the cited Bickle, Rosenberg, Dreyer, Carter and Hawley references disclose conventional testing and debugging mechanisms. However, Applicants respectfully submit that the cited Bickle, Rosenberg, Dreyer, Carter and Hawley references do not enable, nor suggest the subject matter of the invention as recited in the independent claim 1.

Again, this passage does not specifically explain how the language of the claims distinguishes them from the references. In particular, none of the cited portions from any of the references are analyzed or argued, only general statements and allegations of patentability are present in the passage. Specific arguments were found on pages 29 and 41 of the brief and are addressed below.

In response to Appellants' argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so

long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

In the middle of page 29 of the Brief, Appellants essentially argue that "Rosenberg does not teach any 'special hardware instruction', nor any equivalent hardware instruction." However, Table 3.1 on page 41 of Rosenberg shows such "special instructions." And the last sentence at the bottom of page 40 reads: "Table 3.1 shows the format for breakpoint **instructions** on the various **CPU architectures** we will be addressing" [emphasis added]. Since these instructions are used by the CPU, they are clearly interpreted as being hardware instructions. Therefore, Appellants' argument is not persuasive.

At the bottom of page 41 of the Brief, Appellants essentially argue that " The compiler of Carter generates object code; however Applicants respectfully submit that Carter does not teach generating hardware instructions as taught and claimed by Applicants, when the claim term is construed in a manner consistent with the ordinary and customary meaning of the claim term and as taught by Applicants the patent specification where the invention is described." The term "compiler-generated hardware instructions" is found on page 7, line 21, and again on page 8 line 18 of the originally filed specification. There does not appear to be any special definition of the term. Carter teaches that compilers generate object code. See at least column 3 lines 60-63: "The **compiler** 14 includes features typically found in compiler products to process a program comprised of source code 18 and **generate object code** 20, such as the IBM VisualAge for COBOL compiler." [emphasis added] Object code is a term used to refer to machine code that

can be directly executed by the CPU (see Carter column 4 lines 21-22). This code is interpreted as being made of hardware instructions, since it is the object code that tells the CPU what to do. Therefore, Appellants' argument is not persuasive.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

/J. Derek Rutten/
Patent Examiner, Art Unit 2192

Conferees:

Tuan Q. Dam, SPE 2192

/Tuan Q. Dam/

Supervisory Patent Examiner, Art Unit 2192

Wei Zhen, SPE 2191

/Wei Zhen/

Supervisory Patent Examiner, Art Unit 2191